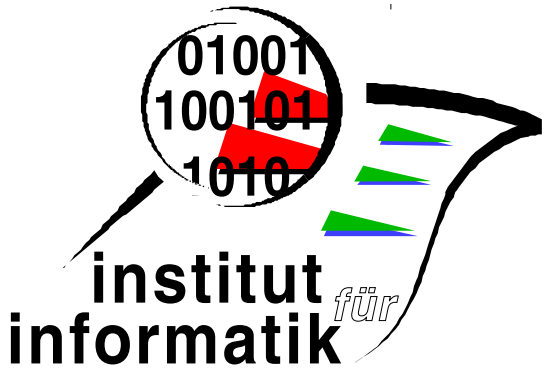
Universität Augsburg

A Graph Algorithmic Framework for the Assembly of Shredded Documents

Fabian Richter, Christian X. Ries, Rainer Lienhart

Report 2011-05 August 2011



Institut für Informatik
D-86135 Augsburg

A GRAPH ALGORITHMIC FRAMEWORK FOR THE ASSEMBLY OF SHREDDED DOCUMENTS

Fabian Richter, Christian X. Ries, Rainer Lienhart

Augsburg University
{richter, ries, lienhart}@informatik.uni-augsburg.de

ABSTRACT

In this paper we propose a framework to address the reassembly of shredded documents. Inspired by the way humans approach this problem we introduce a novel algorithm that iteratively determines groups of fragments that fit together well. We identify such groups by evaluating a set of constraints that takes into account shape- and content-based information of each fragment. Accordingly, we choose the best matching groups of fragments during each iteration and implicitly determine a maximum spanning tree of a graph that represents alignments between the individual fragments. After each iteration we update the graph with respect to additional contextual knowledge. We evaluate the effectiveness of our approach on a dataset of 16 fragmented pages with strongly varying content. The robustness of the proposed algorithm is finally shown in situations in which material is lost.

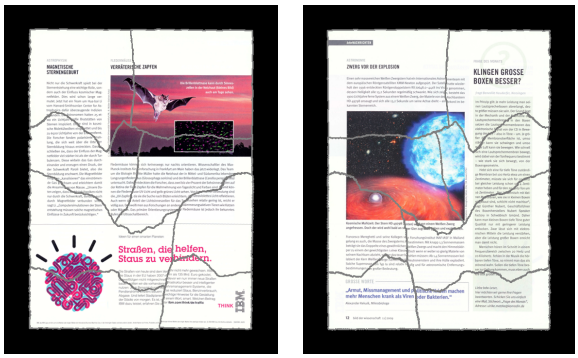
***Index Terms*—** Document assembly, spanning tree algorithm, Kruskal

1. INTRODUCTION

In this paper we present an approach to the problem of automatically reassembling manually shredded documents.

This problem is often faced in the field of forensics. For instance, researchers are currently working on the problem of automatically reassembling a huge amount of documents related to the Stasi which was the secret police of the GDR. Shortly before the Socialist regime of the GDR collapse in 1989, the Stasi destroyed millions of files which contained evidence about their often questionable and illegitimate activities. Some of the files were consigned to paper shredders while others were simply shredded by hand. Today the contents of these files are of high interest to both historians and the German law enforcement. Due to the huge number of documents however it is practically impossible to reconstruct them by hand.

Similar problems are for instance frequently faced by historians dealing with old manuscripts which got destroyed over the years intentionally or by accident. For example, the Historical Archive of the City of Cologne was almost completely destroyed in 2009 when the ground beneath it collapsed and a large amount of valuable historical documents were buried under debris. In the process, most documents presumably got torn apart yet some of these documents might not have been damaged beyond reconstruction.



(a) Front (b) Back

Fig. 1: Example for the simultaneous reconstruction of both sides of a magazine sheet that has been manually shredded into 8 pieces.

In our experiments, we reassemble shredded pages from magazines. However, our approach can be extended for use in applications from the afore mentioned domains. Note that most real-world problems require a framework which appropriately deals with the situation of missing or mixed up document fragments. Thus, we also address these situations in the evaluation section of this paper.

Thus the main contributions of this paper are:

(i) We propose a framework for the iterative assembly of documents by introducing a spanning tree algorithm that adapts its further course of action according to the outcome of preceding iterations.

(ii) Based on a set of constraints we define a measure to quantify the fitness of alignments between groups of fragments. In order to make sound decisions we use shape-, color- and content-based information.

(iii) In our experiments we show that our algorithm provides satisfactory results despite our dataset being diverse in content and color. Besides an evaluation of cases in

978-1-61284-350-6/11/$26.00 ©2011 IEEE

which material is lost, we also demonstrate the simultaneous assembly of multiple pages.

2. RELATED WORKS

Our work was inspired by the work of Cao et al. [1] who automatically reassemble shredded color photos. In contrast to their work we use a different matching strategy based on groups of pieces and constraints between them, such as histograms of oriented gradients (HOG) which were proposed by Dalal [2]. We also employ a novel matching strategy and embed it into a Kruskal-like graph algorithm [3]. For approximating the contours of our shredded pieces we use the Douglas-Peucker algorithm [4].

There are many approaches to the closely related problem of automatically solving jigsaw puzzles. As early as 1964, Freeman et al. [5] for instance tried to assemble a puzzle by defining shape-related features for matching pieces. Radack et al. [6] encoded the borders of jigsaw puzzle pieces in a log polar coordinate system for efficient matching in 1982. In 1998, Chung et al. [7] used shape features to automatically solve jigsaw puzzles but also included color information in the matching process. Contemporary approaches include the works of Nielsen et al. [8] who used an edge-based similarity measure and Sagiroglu et al. [9] who tried to solve puzzles based on texture features.

Another closely related problem is the reconstruction of wall paintings which is addressed by Papaodysseus et al. [10]. They rely on contour shape-based fragment matching.

An overview to the problem of document and artifact reconstruction is given by Kleber et al. in their survey paper [11].

3. ASSEMBLY FRAMEWORK

We introduce an algorithmic approach for the assembly of shredded documents. First, we preprocess each fragment in order to determine its approximate contour. Then, we iteratively rearrange fragments by using a spanning tree algorithm that combines them into clusters of aligned pieces. In each iteration we evaluate a set of constraints among fragments that allow for their accurate alignment. Based on the affine transformations determined in the process, we finally reconstruct the original document.

3.1. Preprocessing

We manually shredded 8 double-sided sheets taken from a scientific magazine as described in detail in section 5.1. Each piece has been scanned, front and back, against a uniformly colored background. In order to account for noise introduced due to this digitalization process we perform a median filtering on each image. Afterwards we subtract the background from the smoothed image by using a color histogram, giving us a binary segmentation. Using the algorithm of Suzuki et al. [12] we finally determine the contour of each fragment from its binary image.

As the exact contour of the i-th fragment is given by a set of pixels P_i that tends to be large in practice, we use the Douglas-Peucker algorithm [4] to determine a subset of support points $S_i = \{s_1^i, s_2^i, \ldots, s_{n_i}^i\} \subseteq P_i$ that allows an accurate yet less complex description of each fragment. By connecting each consecutive pair of support points we obtain a *contour approximation* that makes our approach feasible while maintaining fair performance.

We define each fragment $F_i = (S_i, I_i)$, where S_i is its set of support points and I_i refers to its image content as depicted in figure 2.

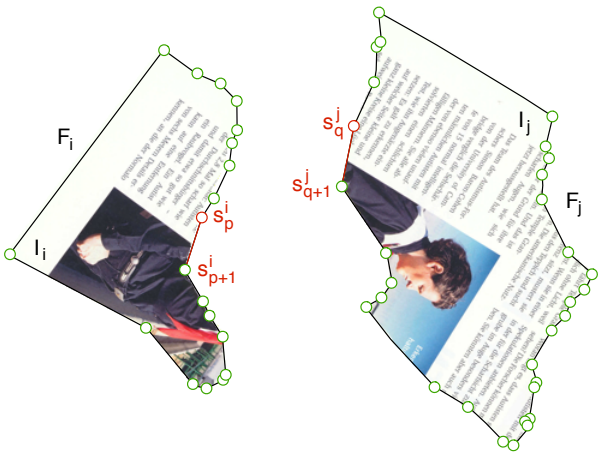


Fig. 2: Line segments constituting the approximate contour of two fragments F_i and F_j. Each segment is defined by a consecutive pair of support points.

3.2. Iterative Merging Algorithm

Given a shredded document we aim to determine a sequence of translations and rotations for each fragment that yields the best assembly result. Therefore we create a weighted graph $\mathcal{G} = (V, E)$ in which each fragment corresponds to a vertex and edges are associated with affine transformations. As discussed in section 4.1, any alignment of two fragments is based on a pair of support points, e.g. (s_p^i, s_q^j). Thus the graph contains multiple edges between each pair of vertices. In order to reduce the number of edges we apply the strategy introduced in [1] to discard pairs of support points that do not match in color and shape. Finally, we weight each edge with respect to an alignment score introduced in section 4.4. Note that edge weights are updated after each iteration. As a consequence, alignments that were seemingly correct in prior iterations due to insufficient contextual evidence may later cause inconsistencies and vice versa. Thereby we adopt the human-like strategy to adjust to situations as they change.

(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$ (e) $t = 5$ (f) $t = 6$ (g) $t = 7$ (h) result

Fig. 3: Combined clusters of each iteration (a)-(g) and the assembled document (h). The red curve along fragment boundaries visualizes the intersecting pixels which constitute the border coincidence constraint defined in section 4.3.

By iteratively choosing the best alignment to combine clusters within each iteration we implicitly determine a spanning tree of $\mathcal{G}$ as in Kruskal's algorithm [3]. Provided as input the support points of each fragment the algorithm returns a sequence of affine transformations Q_i for each fragment F_i. Finally, applying each transformation to its respective fragment yields the assembly of the document.

In the following we describe each step of our proposed algorithm:

ALGORITHM

STEP 1: INITIALIZATION. Initially at time $t = 0$, each fragment F_i is considered a cluster by itself, i.e. $c_i = \{F_i\}$ and $C^{(0)} = \{c_0, c_1, \ldots, c_{|V|}\}$. Each edge is weighted according to its alignment score and Q_i is initialized as an empty sequence of affine transformations.

STEP 2: COMBINING CLUSTERS. Let t be the current iteration. In order to find the best alignment $\hat{a}$ between any pair of clusters, we simply determine the edge with the largest weight that connects two distinct clusters. Let e be this edge and, without loss of generality, let it connect clusters c_i and c_j, $i < j$. By combining both clusters into $\hat{c}_i = c_i \cup c_j$, we align their fragments by affine transformation T as described in section 4.2. As only fragments of cluster c_i are aligned in this step, we append T to their sequence of affine transformation, i.e.

$$\forall k \; F_k \in c_i : \text{append } T \text{ to } Q_k. \tag{1}$$

We obtain the set of clusters for iteration t by replacing the clusters that were combined, i.e.

$$C^{(t)} = \left\{ C^{(t-1)} \backslash \{c_i \cup c_j\} \right\} \cup \hat{c}_i, \tag{2}$$

As we reduce the number of clusters by one during each iteration the algorithm terminates after iteration $t = |V| - 1$. Figure 3 shows intermediate results after each iteration.

Furthermore, we apply a heuristic that removes any pair of support point from $\hat{c}_i$ that became obsolete due to alignment $\hat{a}$. That is, we eliminate support points along coincident borders (illustrated as red curves in figure 3) to reduce the complexity of subsequent iterations.

STEP 3: UPDATING EDGES. As mentioned before, combining two clusters provides additional evidence about the document at hand. Therefore the weight of edges that originate from combined cluster $\hat{c}_i$ need to be updated as described in section 4.4.

3.3. Adaptive Parameter Setting

For each shredded page, fragments vary in size and shape and thus require a different number of support points for accurate contour approximation. However, as this number solely depends on a parameter ϵ used by the Douglas-Peucker algorithm we do not set ϵ to a fixed value in advance. Instead, we automatically adapt it to the given task by running the algorithm multiple times for different values of ϵ and compute a score for each result.

We expect the algorithm to find the correct assembly for some ϵ. For this reason we choose a straightforward score which is guaranteed to prefer a correct assembly over an obviously incorrect one. Thus, we simply count the relative number of non-background pixels of the respective resulting image which fall into a rectangle of the original page format. By maximizing this score the algorithm chooses the best parameter in an unsupervised manner.

4. FRAGMENT CLUSTER ALIGNMENT

In this section we first describe how we align a pair of fragments. Second, we generalize this procedure to situations in which clusters, i.e. groups of fragments, are to be combined.

4.1. Aligning Fragments

In order to align fragment F_i to F_j according to a pair of support points $(s_p^i, s_q^j) \in S_i \times S_j$ we need to apply an affine transform. First, we translate all support points S_i of fragment F_i such that s_p^i coincides with s_q^j. Second, we rotate all support points S_i by angle θ enclosed by the pair of line segments given by (s_p^i, s_{p+1}^i) and (s_q^j, s_{q+1}^j) respectively (see figure 2). Finally, the same translation and rotation is applied to the image content I_i.

4.2. Aligning Clusters

As depicted in figure 4, an alignment between clusters c_i and c_j is defined by an affine transformation between two fragments and a corresponding pair of support points. Without loss of generality, assume that $(s_p^{i_2}, s_q^{j_1})$ from their respective fragments $(F_{i_2}, F_{j_1}) \in c_i \times c_j$ are chosen for the alignment.

In order to align both clusters we first compute the affine transformation T that locally aligns F_{i_2} to F_{j_1}. Afterwards, T is applied to each fragment within c_i.

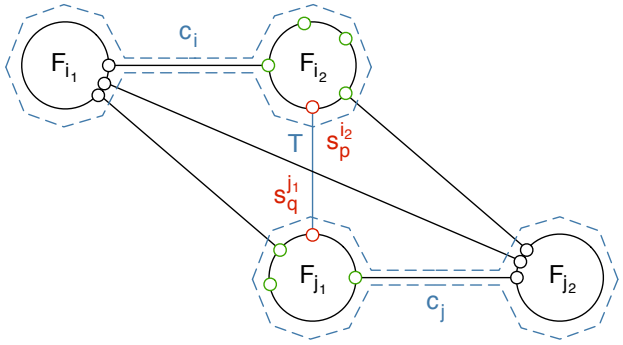


Fig. 4: Clusters are combined by aligning their fragments. Note that we consider any pair of support points, e.g. $(s_p^{i_2}, s_q^{j_1})$ between fragments F_{i_2} and F_{j_1}.

4.3. Constraints

In order to weight the edge representing alignment a between two clusters, we first align their fragments as described in the previous section. Then, we measure the extent to which this aligned group of fragments satisfies a set of constraints. Each constraint is associated with a score and finally concatenated into vector $x^a = [x_k^a]_{k=1..7} \in [0,1]^7$.

BORDER COINCIDENCE. An alignment between groups of fragments results in an overlap between their respective borders. Because of inaccuracies introduced by the contour approximations we allow neighboring pixels to match within a small working band along each side. We determine the *border coincidence* from the number of pixels intersecting within this narrow working band. For an illustrating example see figures 3(a)−(g).

BORDER SHAPE. In many cases a contour segment that is structurally similar to a line is undesirable as it often results in false matches along page boundaries. Therefore, any alignment that produces an overlap which is easily fitted by a straight line is devalued.

BORDER CONNECTIVITY. We also compute the number of connected line segments within the working band of the coincident borders. A high connectivity indicates a continuous border and thus a more reliable alignment.

ABSOLUTE INTERSECTION. Another important fact is that fragments commonly do not have any content overlap. Because of that we compute the *absolute intersection* caused by an alignment defined as the number of overlapping pixels.

RELATIVE INTERSECTION. We further define the *relative intersection* by normalization using the minimum number of pixels covered by fragments of both clusters.

COLOR CONSISTENCY. We determine the *color consistency* at neighboring support points along two matching borders as the average distance between color histograms over the foreground pixels inside a 21×21 region using 16 bins per channel

GRADIENTS. Finally, HOG descriptors [2] are computed over 32×32 pixel regions at each support point along the common border of two aligned clusters. We use 2×2 cells of size 8×8 pixels, resulting in a 124-dimensional vector at each point. We determine a score as the mean distance between vectors of neighboring support points.

4.4. Alignment Score

In this section we introduce a performance measure that rates the quality of an alignment a according to the constraints introduced in section 4.3. Due to a normalization step with respect to all alignments the values of each constraint are restricted to interval $[0,1]$. Finally, based on the normalized vector of constraints we define an *alignment score* as

$$\psi(x^a) = \sum_{k \leq 7} x_k^a. \tag{3}$$

Using this score one can choose the best alignment during iteration t by

$$\hat{a} = \underset{a \in \delta(i,j)}{\arg\max}\, \psi(x^a), \tag{4}$$

where $\delta(i,j)$ refers to the set of alignments between a pair of fragments (F_i, F_j) at time t.

However, as the rotation angle θ for each alignment has been computed based on line segments of approximate contours we allow for slight variations within $[\theta - 5°, \theta + 5°]$ and choose the local optimum that yields the highest alignment score.

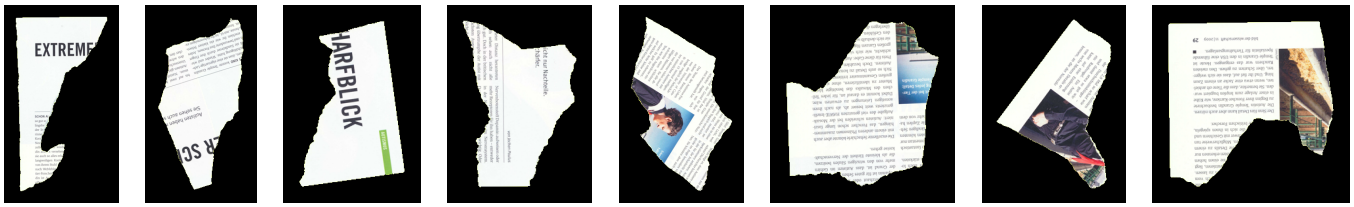
5. EVALUATION

This section describes the experiments we conducted to evaluate our approach and presents some example results.

5.1. Dataset

Our dataset consists of 8 double-sided magazine sheets [1], each shredded manually into 8 pieces, resulting in 16 single-sided pages. We chose sheets varying in color and content. That is, they feature images, tables and large areas of text which we consider more challenging than mere pictures. Figure 5 shows an example of a shredded page.

[1] All sheets were taken from *Bild der Wissenschaft*, 11/2009.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 5: Example for a page shredded into 8 pieces.

5.2. Mean Image

To visualize the overall quality of our approach, we compute a mean image over all resulting reassembled pages. The mean image contains at each pixel position the normalized sum of non-background pixels over the respective pixel position in each resulting image. For example, a value of 1 would be assigned to a pixel position which is covered by a fragment inside of each resulting image and therefore would appear white in the mean image. Since we do not know which pixel positions relate to each other in different result images, we center all result images at the centroid of their non-background pixels.

We can now visualize the overall result on our dataset since the mean image reflects the areas where most of the reassembled fragments are placed by the algorithm. That is, the more alike the mean image is to the original page format, the better the overall performance of the algorithm.

5.3. Assembly of Single-Sided Pages

In our first experiment we evaluate our approach on 16 distinct pages by considering both sides of each sheet, front and back. Our algorithm reassembles all of our pages correctly for several values of ϵ.

Figure 6 shows one example of a reassembled page and the mean image which was computed over all 16 results of documents shredded into 8 pieces.

5.4. Assembly of Double-Sided Pages

We combine the fragments of each sheets's front and back side into mixed sets, each consisting of 16 fragments. Each of these sets is used as input to our algorithm which we made to terminate at two remaining clusters for this task. As all pages were reassembled correctly, we obtained results identical to the results of the reassembly of the single-sided pages. Therefore we obtain the same mean image as the one depicted in figure 6a. Figure 1 shows an example for a simultaneously reassembled front and back side.

We decided to mix a page with its own back for two reasons: First, we consider it slightly more challenging than reassembling two totally different pages, as the contours of the fragments of both sides are flipped versions of each other. Secondly, in a real world application one might not know which side of the fragments belongs to the front of the shredded document. However, since the algorithm is capable of reassembling the respective sides correctly, one simply has to scan both sides of all fragments.

(a) Example result (b) Mean

Fig. 6: Results for the assembly of 16 pages fragmented into 8 pieces each.

5.5. Material Loss

In many real-world applications fragments of shredded documents may be missing. However, the assembly of documents in absence of one or more fragments is a particularly challenging task as it requires an understanding for the document as an entity, i.e. its size and shape. Note that by design our algorithm is completely unsupervised in that it does not introduce any constraint regarding the page format of the shredded document. That is, we do not introduce any bias to maintain an overall page-like shape in cases in which fragments are missing. As a result, the algorithm tends to arrange fragments densely instead of fitting them around gaps (see figure 7).

Nevertheless, we evaluate our algorithm on all possible subsets of 7 out of 8 fragments per page. Surprisingly, it is capable of assembling 71% of these subsets correctly. As can be seen in figure 7 the majority of fragments is still aligned correctly even if the overall assembly fails.

(a) (b)

Fig. 7: Examples of correct (a) and incorrect (b) assembly results in the absence of a single fragment.

(a) Assembly result (b) Mean

Fig. 8: Results obtained from the assembly of 16 pages fragmented into 16 pieces each.

5.6. Increased Number of Fragments

In our final experiment, we doubled the number of fragments by bisecting each of the 8 pieces of each page. Still, our algorithm yields good results, i.e. 15 out of 16 pages were assembled correctly by using two different values of ϵ for our adaptive parameter setting. The respective mean image and some example results are shown in figure 8.

6. CONCLUSION

We have proposed a graph-based approach to reassembling manually shredded documents. Each edge in the graph has been weighted according to the alignment score between its group of fragments. During each iteration of our algorithm we select the edge with the largest weight and determine the corresponding affine transformation. As a result we obtain a sequence of affine transformations for each fragment. Finally we apply each transformation to its respective fragment and obtain a reconstruction of the shredded document. In our evaluation we showed the effectiveness of our approach in different scenarios.

7. REFERENCES

[1] Shengjiao Cao, Hairong Liu, and Shuicheng Yan, "Automated assembly of shredded pieces from multiple photos," in *IEEE International Conference on Multimedia & Expo*, 2010.

[2] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[3] Joseph Kruskal, "On the shortest spanning subtree and the traveling salesman problem," in *Proceedings of the American Mathematical Society*, 1956, vol. 7, pp. 48–50.

[4] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," in *The Canadian Cartographer*, 1973, vol. 10, pp. 112–122.

[5] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," in *IEEE Transactions on Electronic Computers*, 1964, vol. 13, pp. 118–127.

[6] Gerald M. Radack and Norman I. Badler, "Jigsaw puzzle matching using a boundary-centered polar encoding," in *Computer Graphics and Image Processing*, May 1982, vol. 19, pp. 1–17.

[7] Min Gyo Chung, M.M. Fleck, and D.A. Forsyth, "Jigsaw puzzle solver using shape and color," in *Proceedings of the Fourth International Conference on Signal Processing*, 1998, vol. 2, pp. 877–880.

[8] Ture R. Nielsen, Peter Drewsen, and Klaus Hansen, "Solving jigsaw puzzles using image features," in *Pattern Recognition Letters*, New York, NY, USA, October 2008, vol. 29, pp. 1924–1933, Elsevier Science Inc.

[9] M.S. Sagiroglu and A. Ercil, "A texture based matching approach for automated assembly of puzzles," in *18th International Conference on Pattern Recognition*, 2006, vol. 3, pp. 1036–1041.

[10] C. Papaodysseus, T. Panagopoulos, M. Exarhos, C. Triantafillou, D. Fragoulis, and C. Doumas, "Contour-shape based reconstruction of fragmented, 1600 bc wall paintings," in *IEEE Transactions on Signal Processing*, June 2002, vol. 50, pp. 1277–1288.

[11] Florian Kleber and Robert Sablatnig, "A survey of techniques for document and archaeology artefact reconstruction," in *International Conference on Document Analysis and Recognition*, Los Alamitos, CA, USA, 2009, vol. 0, pp. 1061–1065, IEEE Computer Society.

[12] S. Suzuki and K. Abe, "Topological structural analysis of digital binary images by border following," in *CVGIP*, 1985, vol. 30, pp. 32–46.